Chih Hsuan Huang
ID:934554197
huanchih@oregonstate.edu
CS 575

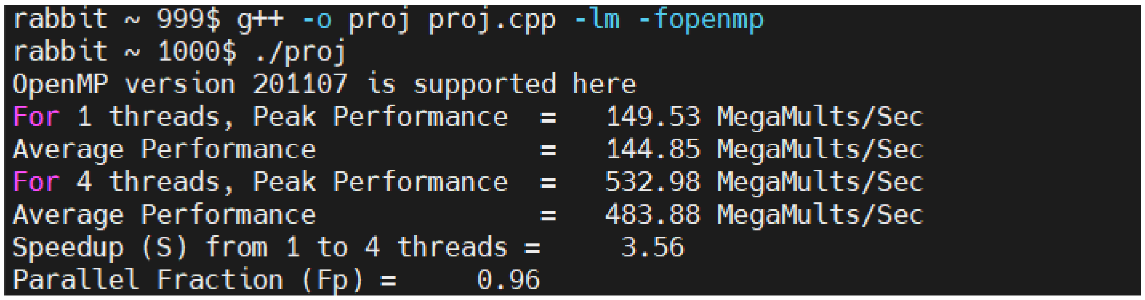
Project 0 - Simple OpenMP Experiment

1. Tell what machine you ran this on

   CPU: Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
   Memory: 64GB
   Machine: rabbit.engr.oregonstate.edu
   Array size: 16384

2. What performance results did you get?

Number of Threads	Peak Performance	Average Performance
Using 1 thread	149.53 MegaMults/Sec	144.85 MegaMults/Sec
Using 4 threads	532.98 MegaMults/Sec	483.88 MegaMults/Sec



```
rabbit ~ 999$ g++ -o proj proj.cpp -lm -fopenmp
rabbit ~ 1000$ ./proj
OpenMP version 201107 is supported here
For 1 threads, Peak Performance  =   149.53 MegaMults/Sec
Average Performance              =   144.85 MegaMults/Sec
For 4 threads, Peak Performance  =   532.98 MegaMults/Sec
Average Performance              =   483.88 MegaMults/Sec
Speedup (S) from 1 to 4 threads =     3.56
Parallel Fraction (Fp) =     0.96
```

3. What was your 4-thread-to-one-thread speedup?

   Speedup, S = (Performance with four threads) / (Performance with one thread)

   Speedup, S = P4 / P1 = 532.98 / 149.53 = 3.56

4. Why do you think it is behaving this way?

In the experiment, when increasing the number of threads from 1 to 4, the observed performance improvement was approximately 3.56 times, which is less than the ideal 4 times. This is mainly because when the number of added threads exceeds the number of cores in the processor, the performance improvement is no longer linear because the cores must switch between multiple threads, adding additional overhead. Furthermore, according to Amdahl's law, the maximum speedup of a program is limited by the portion of it that can be parallelized. Even if some programs can be perfectly parallelized, some still need to be executed serially, which limits the overall performance improvement.

5. What was your Parallel Fraction, Fp?
float Fp = (4./3.)*(1. - (1./S));
Fp = (4./3.)*(1. - (1./3.56));Fp =
0.96